

1 METHODS AND APPARATUS FOR MODIFYING
2 PROGRAMS STORED IN READ ONLY MEMORY

3
4 BACKGROUND OF THE INVENTION
5

6 1. Field of the Invention

7 The invention relates to embedded systems. More
8 particularly, the invention relates to methods and apparatus for
9 modifying embedded system programs stored in read only memory
10 (ROM).

11
12 2. State of the Art

13 Digital computing systems are typically known to include
14 hardware and software. Software is typically stored on a writable
15 medium such as a magnetic disk. As most computer users know,
16 software often contains errors or mistakes which prevent it from
17 functioning properly. Software vendors often publish updates or
18 "patches" to correct errors in software. An update is typically a
19 new complete version of the original software which is intended to
20 replace the entire original program. The original software is
21 erased from the writable medium and the replacement software is
22 written to the writable medium in place of the original. A patch
23 is a program which is run to modify the original software. The
24 patch program finds the portion(s) of the software which need to
25 be replaced and overwrites them with replacement code.

1 Today, digital computing systems are ubiquitous and often
2 unseen. These systems are designed to perform specialized
3 functions such as controlling the operation of an appliance, a
4 motor vehicle, or a computer peripheral device such as a modem or
5 a printer. Such computing systems are usually referred to as
6 "embedded systems". They include a microprocessor and software
7 which is typically stored on a read only memory (ROM). ROM is
8 advantageous because it is non-volatile, small, inexpensive, and
9 energy efficient. Program code stored on ROM is usually called
10 "firmware" rather than software because once it is stored on ROM,
11 the code cannot be modified. The code takes on the quality of
12 hardware in the sense that in order to change it, the physical ROM
13 device must be replaced. Indeed, many embedded systems have
14 "socketed" ROM so that the ROM can be easily unplugged and
15 replaced with a new ROM if the program needs to be changed.
16 However, that is not always practical or convenient.

17

18 One known method for alteration of firmware programs in ROM-
19 based systems is disclosed in U.S. Patent Number 4,607,332, issued
20 on August 19, 1986 to Goldberg. The method allows for dynamic
21 alteration of ROM programs with the aid of random access memory
22 (RAM) and through the use of the standard linkages associated with
23 subroutine calls in the system processor. When each ROM-based
24 routine is written, one program statement is a call to a ROM-
25 located processing routine which searches a RAM-located data

1 structure. If there exists a correspondence between information
2 passed on the call to the processing routine and certain elements
3 of the data structure, a RAM-based program is substituted for the
4 ROM-based program. After adjusting the processor to the states
5 just after the call to the ROM-based program, the processing
6 routine effects a transfer to the replacement routine in RAM. The
7 location of this replacement routine is also found in the data
8 structure. The main disadvantage of the method proposed by
9 Goldberg is that it is inefficient. It uses up too much space in
10 ROM and RAM. It also requires an external compiler/interpreter.
11 Since the processing function is ROM-based, it is limited in scope
12 and potential to be modified. The ROM-based function also
13 presents excessive processing overhead since it must perform a
14 search for possible replacement code even when no replacement code
15 exists

17 Another system for altering ROM-based firmware is disclosed
18 in U.S. Patent Number 5,740,351, issued on April 14, 1998 to
19 Kasten. The system relies on an "extensible interpreter", i.e. a
20 modified FORTH kernel and a plurality of customizable call outs
21 (CCOs). CCOs are present in the ROM-based program. When a CCO is
22 encountered during execution of the program, the modified FORTH
23 kernel takes control and looks for the called function or
24 parameter. If it is found (in RAM or in ROM), it is executed or
25 fetched and the result is returned by the modified FORTH kernel to

1 the next instruction in the ROM program. The system is primarily
2 intended for interactive use with a dumb terminal during
3 "debugging" of the ROM-based program. CCOs in RAM must be defined
4 using the modified FORTH kernel. The main advantage of Kasten
5 over Goldberg is that Kasten does not require an external
6 compiler/interpreter. Disadvantages of Kasten are that it is
7 limited to modifications made using the FORTH programming language
8 and it is inefficient, requiring that a substantial part of ROM be
9 devoted to the modified FORTH kernel.

0

1 Still another system for altering software in embedded
2 systems is disclosed in U.S. Patent Number 5,901,225, issued on
3 May 4, 1999 to Ireton et al. The system includes an embedded
4 system device coupled to an external memory. The device includes
5 a non-alterable memory, including firmware, coupled to a
6 processor. The device further includes a relatively small amount
7 of patch RAM within the device also coupled to the processor.
8 Patches are loaded from the external memory into the patch RAM.
9 The device further includes a means for determining if one or more
20 patches are to be applied. If the device detects a patch to be
21 applied, the system loads the patch from the external memory into
22 the patch RAM. The device also includes a breakpoint register.
23 When the value of the program counter of the processor equals the
24 value in the breakpoint register, a patch insertion occurs, i.e.,
25 the processor deviates from executing firmware to executing patch

1 instructions. The system described by Ireton et al. is quite
2 complex.

3

4 SUMMARY OF THE INVENTION

5

6 It is therefore an object of the invention to provide methods
7 and apparatus for modifying firmware code stored in a read only
8 memory.

9

0 It is also an object of the invention to provide methods and
1 apparatus for modifying firmware code stored in a read only memory
2 which make efficient use of RAM.

3

4 It is another object of the invention to provide methods and
5 apparatus for modifying firmware code stored in a read only memory
6 which are relatively simple in architecture.

17

18 It is also an object of the invention to provide methods and
19 apparatus for modifying firmware code stored in a read only memory
20 which do not require any decision making elements in ROM to
21 support future code modifications

22

23 It is another object of the invention to provide methods and
24 apparatus for modifying firmware code stored in a read only memory

1 which do not require any processor specific or processor dependent
2 elements.

3

4 It is still another object of the invention to provide
5 methods and apparatus for modifying firmware code stored in a read
6 only memory which are applicable to any programming language.

7

8 It is also an object of the invention to provide methods and
9 apparatus for modifying firmware code stored in a read only memory
10 which do not limit the scope of future code modifications.

11

12 It is another object of the invention to provide methods and
13 apparatus for modifying firmware code stored in a read only memory
14 which incur minimum processing overhead.

15

16 It is still another object of the invention to provide
17 methods and apparatus for modifying firmware code stored in a read
18 only memory which minimize the size of ROM code segments needed to
19 be replaced to fix an error.

20

21 In accord with these objects which will be discussed in
22 detail below, the apparatus of the invention includes an embedded
23 device having program code stored in ROM and an on-board or
24 external RAM for storing modified code segments. The methods of
25 the invention include structuring the ROM-based firmware so that a

1 RAM-based function is called prior to each potentially modifiable
2 code segment. Prior to modifying the firmware, a dummy function
3 is stored in RAM so that every call to RAM is simply returned to
4 ROM. When a segment of code is to be modified, a replacement is
5 stored in RAM and indexed by the return address of the function
6 call. The system of the present invention is efficient as it uses
7 very little RAM. It does not require any ROM-based decision
8 making elements; and it is not limited to a particular programming
9 language or processor. The system of the invention is most
10 suitable for use in a computer peripheral which communicates with
11 a higher level controller, e.g. a personal computer, from which
12 replacement code can be downloaded. Alternatively, replacement
13 code in RAM can be loaded by a small bootstrap program (i.e. a
14 run-time system initialization program) stored in replaceable
15 external ROM.

16

17 Additional objects and advantages of the invention will
18 become apparent to those skilled in the art upon reference to the
19 detailed description taken in conjunction with the provided
20 figures.

21

22

BRIEF DESCRIPTION OF THE DRAWINGS

23

24 Figure 1 is a simplified block diagram of an embedded system
25 according to the invention;

1

2 Figure 2 is a schematic illustration of the organization of
3 ROM-based code according to the invention;

4

5 Figure 3 is a schematic illustration of the dummy function in
6 RAM according to the invention;

7

8 Figure 4 is a schematic illustration of the replacement code
9 in RAM indexed to return address; and

10

11 Figure 5 is a simplified flowchart of the operation of the
12 invention.

13

14 DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

15

16 Referring now to Figure 1, an embedded system 10 according to
17 the invention includes a processor 12 which is coupled to a
18 firmware ROM 14 and a RAM 16. As illustrated in Figure 1, the RAM
19 16 is ideally much smaller than the ROM 14, for example
20 approximately one one hundredth the size of the ROM.

21

22 Turning now to Figure 2, according to the invention, the
23 firmware stored in the ROM 14 is divided into code segments, e.g.
24 18a-18d, 20a-20d, 22a-22d, and 24a-24d. Prior to the start of
25 each code segment, a call function with a (calling and) return

1 address is placed, e.g. at 18, 20, 22, and 24. The locations of
2 the call functions are preferably organized for the most efficient
3 use of RAM. It might seem practical to locate each call function
4 at each logical break in the code, e.g. at the start of each new
5 routine. However, in the case of very long routines, it can be
6 more efficient to place call functions equally spaced throughout
7 the routine. This will be better understood after considering the
8 code listings below. Essentially, whenever replacement code is
9 used, all of the code from the call function until the corrected
10 code is reached is replaced. Thus if there are one thousand lines
11 of code between call functions and only the six hundredth line of
12 that code needs to be changed, six hundred lines will be replaced
13 nonetheless. If the call functions were spaced every one hundred
14 lines throughout the thousand line routine, no more than one
15 hundred lines would need to be replaced. Thus, a balance must be
16 struck between the number of call functions placed throughout the
17 ROM-based code and the amount of RAM needed to make code
18 replacements.

19

20 Figure 3 illustrates the structure of RAM 16 when none of the
21 code segments in ROM is to be replaced. A dummy function 26,
22 including instruction lines 26a-26d resides in RAM 16. Whenever a
23 call function (18, 20, 22, 24 in Figure 2) is executed, the dummy
24 function 26 responds by returning program execution to the return

1 address (which is usually the next line after the calling function).

2

3 Figure 4 illustrates the structure of RAM 16 when some of the
4 code segments in ROM are to be replaced. In this case, the Return
5 in the Dummy Function is replaced with a branch to the
6 corresponding Patch Center Function. According to the presently
7 preferred embodiment, a lookup table 28, including a "patch center
8 function" (line 1 of the RAM table) provides the functionality
9 necessary to determine whether a code segment has a replacement in
10 RAM. Operation of the patch center function is described in more
11 detail below with reference to Figure 5 and Code Listing 2. The
12 lookup table is illustrated schematically at lines 2-4 of the RAM
13 table, line 2 being the start of table (SOT) and line 4 being the
14 end of table (EOT). Each entry in the lookup table refers to a
15 return address (RA) and a patch address (PA). For example, the
16 first entry refers to a return address of X+1 and a patch address
17 of 5. This signifies that the code being replaced is a
18 replacement (shown at 30 in Figure 4) for code segment X; the
19 replacement code begins at line 5 in the RAM table and upon
20 execution of the replacement code, execution continues at an
21 address specified in the replacement code, typically the next
22 segment after X. It is possible to replace only a portion of a
23 code segment and return to a line within the original code segment
24 to continue execution. As will be seen below in Code Listing 2,
25 it is necessary to replace a contiguous set of code lines from the

1 call function forward. Similar entries are indicated for code
2 segment Y and Z replacements which are illustrated at 32 and 34 in
3 Figure 4.

4

5 Referring now to Figure 5, when the patch center function is
6 called at 100, the lookup table entries are scanned at 102. If a
7 return address match is found at 104, program execution is
8 directed to the patch address at 106. The replacement code is
9 executed at 108, and program execution is returned to a ROM
10 address specified in the patch code at 110. So long as the end of
11 the table has not been reached as determined at 112, the patch
12 center function looks for a table entry for replacement code.

13 When it is determined that the end of the table has been reached,
14 program execution returns to the return address in ROM at 114. As
15 mentioned above, if there is no replacement code in RAM, program
16 execution returns to the return address specified by the call
17 function which also acts as an index to the lookup table. If
18 replacement code is executed, the return address is specified by
19 the replacement code.

20

21 The operation of the invention will be better understood with
22 reference to the following Code Listing 1 and Code Listing 2 which
23 illustrate the structure of the code in ROM and RAM respectively.

```
1
2 ****
3 * Code Listing 1 (ROM Segment)
4 ****
5
6 Code_in Rom:
7     Patch_Call1()           ; Function call
8
9
10
11     Patch_Call1()
12
13
14
15     Patch_Call2()
16 ROM_Return_Addr2:           ; Return Address (RA)
17     I = I + 10
18     K + K - 55             ; Segment code starts
19 ROM_Address2:              ; Need modification !
20     M = M + 88
21
22
23     Patch_Call2()          ; segment code ends
24     J = J - 15
25
26
27     Patch_Call2()
28
29
30
31     Patch_Call3()
32
33
34
35     Patch_Call3()
36
```

Code Listing 1 (ROM Segment)

39

40 Referring to Code Listing 1, a code segment is shown with
41 seven patch calls at lines 7, 11, 15, 23, 27, 31, and 35. Most of
42 the code is not shown but is represented by dots, i.e. at lines 8-
43 10, 12-14, 21, 22, 25, 26, 28-30, and 32-34. The code which will
44 be modified follows patch call2() at line 15.

```

1 ****
2 * Code Listing 2 (RAM Segment)
3 ****
4 Code_in_RAM:
5
6 Patch_Call1
7     Return ; Dummy function
8 ; Return to ROM Code
9 Patch_Call2:
10    Go to Patch_Center 2 ; Dummy Function
11 ; Replace Return here !
12 Patch_Call3:
13     Return ; Dummy Function
14 ; Return to ROM Code
15 .
16 .
17 .
18 Patch_Center2: ; Patch Center Function
19
20     Save_Context() ; Save necessary registers
21
22     Address_Match = Patch_Address_Table_Search() ; Lookup table searching
23
24
25     if (Address_Match == YES)
26         Go to Patch_Code2 ; Have replacement code
27     else
28         Restore_Context() ; Restore the registers
29         Return ; Return to ROM code
30
31 Patch_Center2_Lookup_SOT: ; Start of Lookup Table
32     ROM_Return_Addr2 ; Return Address (RA)
33     Patch_Code2 ; Patch Address (PA)
34
35 .
36 .
37 Patch_Center_Lookup_EOT: ; End of Lookup Table
38
39 Patch_Code2:
40     I = I + 10 ; Patch Address (PA)
41     K = K - 99 ; Replacement code
42     Restore_Context() ; Replace K = K - 55 in ROM
43     Return to ROM_Address2 ; Restore the registers
44
45
46
47

```

Code Listing 2 (RAM Segment)

48 Turning now to Code Listing 2, lines 6-13 represent the dummy
49 functions. As shown, patch calls 1 and 3 are dummy functions
50 which return the program to ROM. Patch call 2 directs the program

1 to patch center 2 which begins at line 18. Patch center 2 saves
2 the context and checks the address match. The address match
3 lookup table is illustrated at lines 31-37. If the address
4 matches, the program is directed to patch code 2 which starts at
5 line 39. If the address does not match, context is restored and
6 the program is returned to ROM. As shown at lines 39-43, patch
7 code 2 is designed to replace two lines of code in code listing 1,
8 i.e. replace lines 17 and 18 of code listing 1 with lines 34 and
9 35 of code listing 2. After executing lines 40 and 41 of code
10 listing 2, context is restored at line 42 and the program is
11 returned to ROM at 43. The return address is indicated in code
12 listings 1 and 2 as ROM_Address2. The actual return address is
13 derived from knowledge of the code in ROM. It should be noted
14 that line 40 of code listing 2 is identical to line 17 of code
15 listing 1. Nevertheless, it is replaced because, as mentioned
16 above, the function calls according to the invention allow and
17 require replacement of all code from the function call through the
18 corrected code.

19

20 There have been described and illustrated herein methods and
21 apparatus for modifying program code stored in ROM. While
22 particular embodiments of the invention have been described, it is
23 not intended that the invention be limited thereto, as it is
24 intended that the invention be as broad in scope as the art will
25 allow and that the specification be read likewise. It will

1 therefore be appreciated by those skilled in the art that yet
2 other modifications could be made to the provided invention
3 without deviating from its spirit and scope as so claimed.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
33510
33511
33512
33513
33514
33515
33516
33517
33518
33519
33520
33521
33522
33523
33524
33525
33526
33527
33528
33529
33530
33531
33532
33533
33534
33535
33536
33537
33538
33539
33540
33541
33542
33543
33544
33545
33546
33547
33548
33549
33550
33551
33552
33553
33554
33555
33556
33557
33558
33559
33560
33561
33562
33563
33564
33565
33566
33567
33568
33569
33570
33571
33572
33573
33574
33575
33576
33577
33578
33579
33580
33581
33582
33583
33584
33585
33586
33587
33588
33589
33590
33591
33592
33593
33594
33595
33596
33597
33598
33599
335100
335101
335102
335103
335104
335105
335106
335107
335108
335109
335110
335111
335112
335113
335114
335115
335116
335117
335118
335119
335120
335121
335122
335123
335124
335125
335126
335127
335128
335129
335130
335131
335132
335133
335134
335135
335136
335137
335138
335139
335140
335141
335142
335143
335144
335145
335146
335147
335148
335149
335150
335151
335152
335153
335154
335155
335156
335157
335158
335159
335160
335161
335162
335163
335164
335165
335166
335167
335168
335169
335170
335171
335172
335173
335174
335175
335176
335177
335178
335179
335180
335181
335182
335183
335184
335185
335186
335187
335188
335189
335190
335191
335192
335193
335194
335195
335196
335197
335198
335199
335200
335201
335202
335203
335204
335205
335206
335207
335208
335209
335210
335211
335212
335213
335214
335215
335216
335217
335218
335219
335220
335221
335222
335223
335224
335225
335226
335227
335228
335229
335230
335231
335232
335233
335234
335235
335236
335237
335238
335239
335240
335241
335242
335243
335244
335245
335246
335247
335248
335249
335250
335251
335252
335253
335254
335255
335256
335257
335258
335259
335260
335261
335262
335263
335264
335265
335266
335267
335268
335269
335270
335271
335272
335273
335274
335275
335276
335277
335278
335279
335280
335281
335282
335283
335284
335285
335286
335287
335288
335289
335290
335291
335292
335293
335294
335295
335296
335297
335298
335299
335300
335301
335302
335303
335304
335305
335306
335307
335308
335309
335310
335311
335312
335313
335314
335315
335316
335317
335318
335319
335320
335321
335322
335323
335324
335325
335326
335327
335328
335329
335330
335331
335332
335333
335334
335335
335336
335337
335338
335339
335340
335341
335342
335343
335344
335345
335346
335347
335348
335349
335350
335351
335352
335353
335354
335355
335356
335357
335358
335359
335360
335361
335362
335363
335364
335365
335366
335367
335368
335369
335370
335371
335372
335373
335374
335375
335376
335377
335378
335379
335380
335381
335382
335383
335384
335385
335386
335387
335388
335389
335390
335391
335392
335393
335394
335395
335396
335397
335398
335399
335400
335401
335402
335403
335404
335405
335406
335407
335408
335409
335410
335411
335412
335413
335414
335415
335416
335417
335418
335419
335420
335421
335422
335423
335424
335425
335426
335427
335428
335429
335430
335431
335432
335433
335434
335435
335436
335437
335438
335439
335440
335441
335442
335443
335444
335445
335446
335447
335448
335449
335450
335451
335452
335453
335454
335455
335456
335457
335458
335459
335460
335461
335462
335463
335464
335465
335466
335467
335468
335469
335470
335471
335472
335473
335474
335475
335476
335477
335478
335479
335480
335481
335482
335483
335484
335485
335486
335487
335488
335489
335490
335491
335492
335493
335494
335495
335496
335497
335498
335499
335500
335501
335502
335503
335504
335505
335506
335507
335508
335509
335510
335511
335512
335513
335514
335515
335516
335517
335518
335519
335520
335521
335522
335523
335524
335525
335526
335527
335528
335529
335530
335531
335532
335533
335534
335535
335536
335537
335538
335539
335540
335541
335542
335543
335544
335545
335546
335547
335548
335549
335550
335551
335552
335553
335554
335555
335556
335557
335558
335559
335560
335561
335562
335563
335564
335565
335566
335567
335568
335569
335570
335571
335572
335573
335574
335575
335576
335577
335578
335579
335580
335581
335582
335583
335584
335585
335586
335587
335588
335589
335590
335591
335592
335593
335594
335595
335596
335597
335598
335599
3355100
3355101
3355102
3355103
3355104
3355105
3355106
3355107
3355108
3355109
3355110
3355111
3355112
3355113
3355114
3355115
3355116
3355117
3355118
3355119
3355120
3355121
3355122
3355123
3355124
3355125
3355126
3355127
3355128
3355129
3355130
3355131
3355132
3355133
3355134
3355135
3355136
3355137
3355138
3355139
3355140
3355141
3355142
3355143
3355144
3355145
3355146
3355147
3355148
3355149
3355150
3355151
3355152
3355153
3355154
3355155
3355156
3355157
3355158
3355159
3355160
3355161
3355162
3355163
3355164
3355165
3355166
3355167
3355168
3355169
3355170
3355171
3355172
3355173
3355174
3355175
3355176
3355177
3355178
3355179
3355180
3355181
3355182
3355183
3355184
3355185
3355186
3355187
3355188
3355189
3355190
3355191
3355192
3355193
3355194
3355195
3355196
3355197
3355198
3355199
3355200
3355201
3355202
3355203
3355204
3355205
3355206
3355207
3355208
3355209
3355210
3355211
3355212
3355213
3355214
3355215
3355216
3355217
3355218
3355219
3355220
3355221
3355222
3355223
3355224
3355225
3355226
3355227
3355228
3355229
3355230
3355231
3355232
3355233
3355234
3355235
3355236
3355237
3355238
3355239
3355240
3355241
3355242
3355243
3355244
3355245
3355246
3355247
3355248
3355249
3355250
3355251
3355252
3355253
3355254
3355255
3355256
3355257
3355258
3355259
3355260
3355261
3355262
3355263
3355264
3355265
3355266
3355267
3355268
3355269
3355270
3355271
3355272
3355273
3355274
3355275
3355276
3355277
3355278
3355279
3355280
3355281
3355282
3355283
3355284
3355285
3355286
3355287
3355288
3355289
3355290
3355291
3355292
3355293
3355294
3355295
3355296
3355297
3355298
3355299
3355300
3355301
3355302
3355303
3355304
3355305
3355306
3355307
3355308
3355309
3355310
3355311
3355312
3355313
3355314
3355315
3355316
3355317
3355318
3355319
3355320
3355321
3355322
3355323
3355324
3355325
3355326
3355327
3355328
3355329
3355330
3355331
3355332
3355333
3355334
3355335
3355336
3355337
3355338
3355339
3355340
3355341
3355342
3355343
3355344
3355345
3355346
3355347
3355348
3355349
3355350
3355351
3355352
3355353
3355354
3355355
3355356
3355357
3355358
3355359
3355360
3355361
3355362
3355363
3355364
3355365
3355366
3355367
3355368
3355369
3355370
3355371
3355372
3355373
3355374
3355375
3355376
3355377
3355378
3355379
3355380
3355381
3355382
3355383
3355384
3355385
3355386
3355387
3355388
3355389
3355390
3355391
3355392
3355393
3355394
3355395
3355396
3355397
3355398
3355399
3355400
3355401
3355402
3355403
3355404
3355405
3355406
3355407
3355408
3355409
3355410
3355411
3355412
3355413
3355414
3355415
3355416
3355417
3355418
3355419
3355420
3355421
3355422
3355423
3355424
3355425
3355426
3355427
3355428
3355429
3355430
3355431
3355432
3355433
3355434
3355435
3355436
3355437
3355438
3355439
3355440
3355441
3355442
3355443
3355444
3355445
3355446
3355447
3355448
3355449
3355450
3355451
3355452
3355453
3355454
3355455
3355456
3355457
3355458
3355459
3355460
3355461
3355462
3355463
3355464
3355465
3355466
3355467
3355468
3355469
3355470
3355471
3355472
3355473
3355474
3355475
3355476
3355477
3355478
3355479
3355480
3355481
3355482
3355483
3355484
3355485
3355486
3355487
3355488
3355489
3355490
3355491
3355492
3355493
3355494
3355495
3355496
3355497
3355498
3355499
3355500
3355501
3355502
3355503
3355504
3355505
3355506
3355507
3355508
3355509
3355510
3355511
3355512
3355513
3355514
3355515
3355516
3355517
3355518
3355519
3355520
3355521
3355522
3355523
3355524
3355525
3355526
3355527
3355528
3355529
3355530
3355531
3355532
3355533
3355534
3355535
3355536
3355537
3355538
3355539
3355540
3355541
3355542
3355543
3355544
3355545
3355546
3355547
3355548
3355549
33555410
33555411
33555412
33555413
33555414
33555415
33555416
33555417
33555418
33555419
33555420
33555421
33555422
33555423
33555424
33555425
33555426
33555427
33555428
33555429
33555430
33555431
33555432
33555433
33555434
33555435
33555436
33555437
33555438
33555439
33555440
33555441
33555442
33555443
33555444
33555445
33555446
33555447
33555448
33555449
33555450
33555451
33555452
33555453
33555454
33555455
33555456
33555457
33555458
33555459
33555460
33555461
33555462
33555463
33555464
33555465
33555466
33555467
33555468
33555469
33555470
33555471
33555472
33555473
33555474
33555475
33555476
33555477
33555478<br